Programming - Part I

Ricard Solé Casas

June 13, 2017

Foreword

The source code for this report and app can be found on Github[1]. The live version of the app itself is online at http://quiz.rsole.me.

Some of the decisions taken in building this app do not follow the original suggested guidelines. The UI is a web frontend, but all the business logic is handled by the server-side via Java.

Declaration

I confirm that the submitted coursework is my own work and that all material attributed to others (whether published or unpublished) has been clearly identified and fully acknowledged and referred to original sources. I agree that the College has the right to submit my work to the plagiarism detection service. TurnitinUK for originality checks.

Acknowledgements

I'd like to thank my partner Shannon for her continued support and challenges that help me grow, both professionally and personally. I would also like to thank all of you who also helped me get here.

---

[1] https://github.com/rcsole/coursework-java

Contents

Chapter 1

Overview

Project is setup using the Java Play[1] framework to aid with following the MVC[2] pattern.

1.1 Models

1.1.1 Option

1.1.2 Question

1.1.3 QuizResult

1.2 Services

1.2.1 QuizService

1.3 Controllers

1.3.1 QuizResultsController

1.3.2 QuizzesController

---

[1]https://playframework.com
[2]https://www.wikiwand.com/en/Model%E2%80%93view%E2%80%93controller

Chapter 2

Test plan

Test	Method	Expected	Actual	Evidence
Selecting 10 creates a quiz with 10 questions	Click 10	There will be 10 questions	As expected	See figure 2.1
Selecting 20 creates a quiz with 20 questions	Click 20	There will be 20 questions	As expected	See figure 2.2
Selecting 30 creates a quiz with 30 questions	Click 30	There will be 30 questions	As expected	See figure 2.3
Cross on top left takes user back to quiz creation	Click x	Quiz will go back to form	As expected	See gif[1]
Selecting an option brings the next question up	Select an option	Next question will come up	As expected	See gif[2]
Skipping will send the question to the end	Skip a question	The question will be skipped and asked again at the end of the quiz	As expected	See gif[3]
When the timer runs out the quiz gets submitted	Wait for timer to run out	The quiz gets submitted	As expected	See gif[4]
Score is displayed as a percentage	Finish the quiz	The score is displayed upon finishing the quiz	As expected	See figure 2.4

---

[1] http://www.giphy.com/gifs/3ohzdEZt9v5mq8oAsE
[2] http://www.giphy.com/gifs/3og0IMCTcnr7RFvaaQ
[3] http://www.giphy.com/gifs/l1BgSVSrual0DUzDi
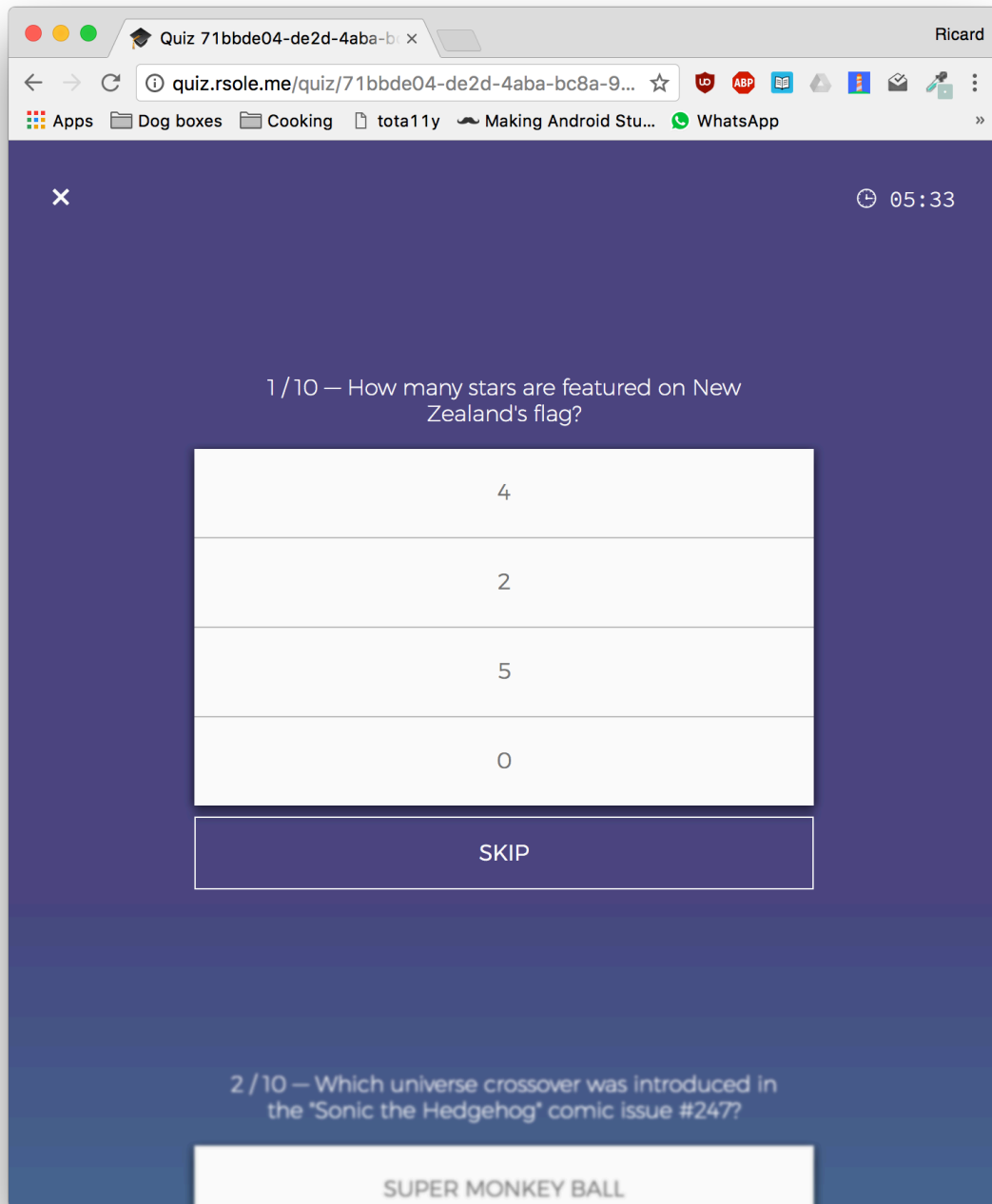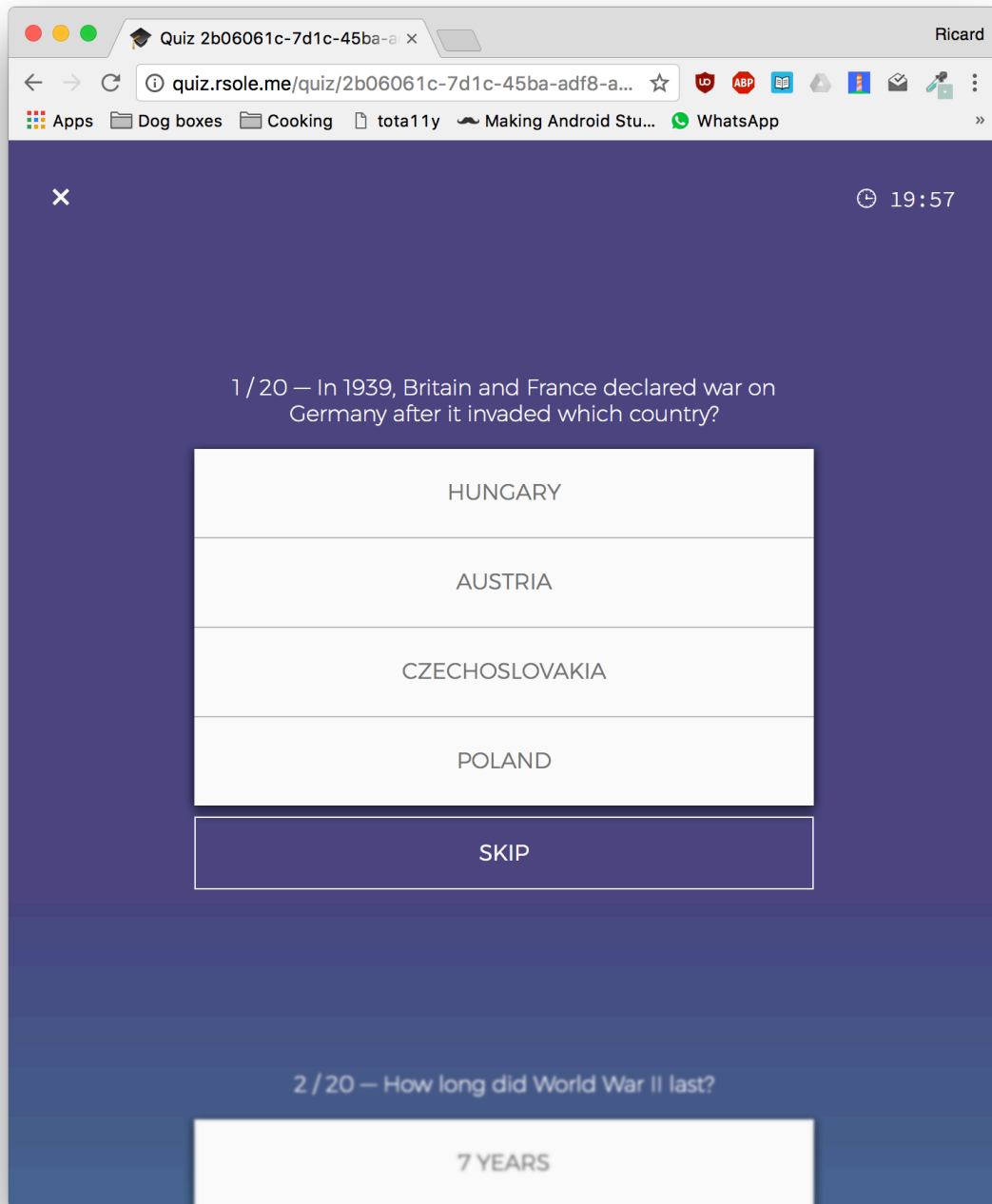[4] http://www.giphy.com/gifs/l0Iy8yTqqq5BCf1yU

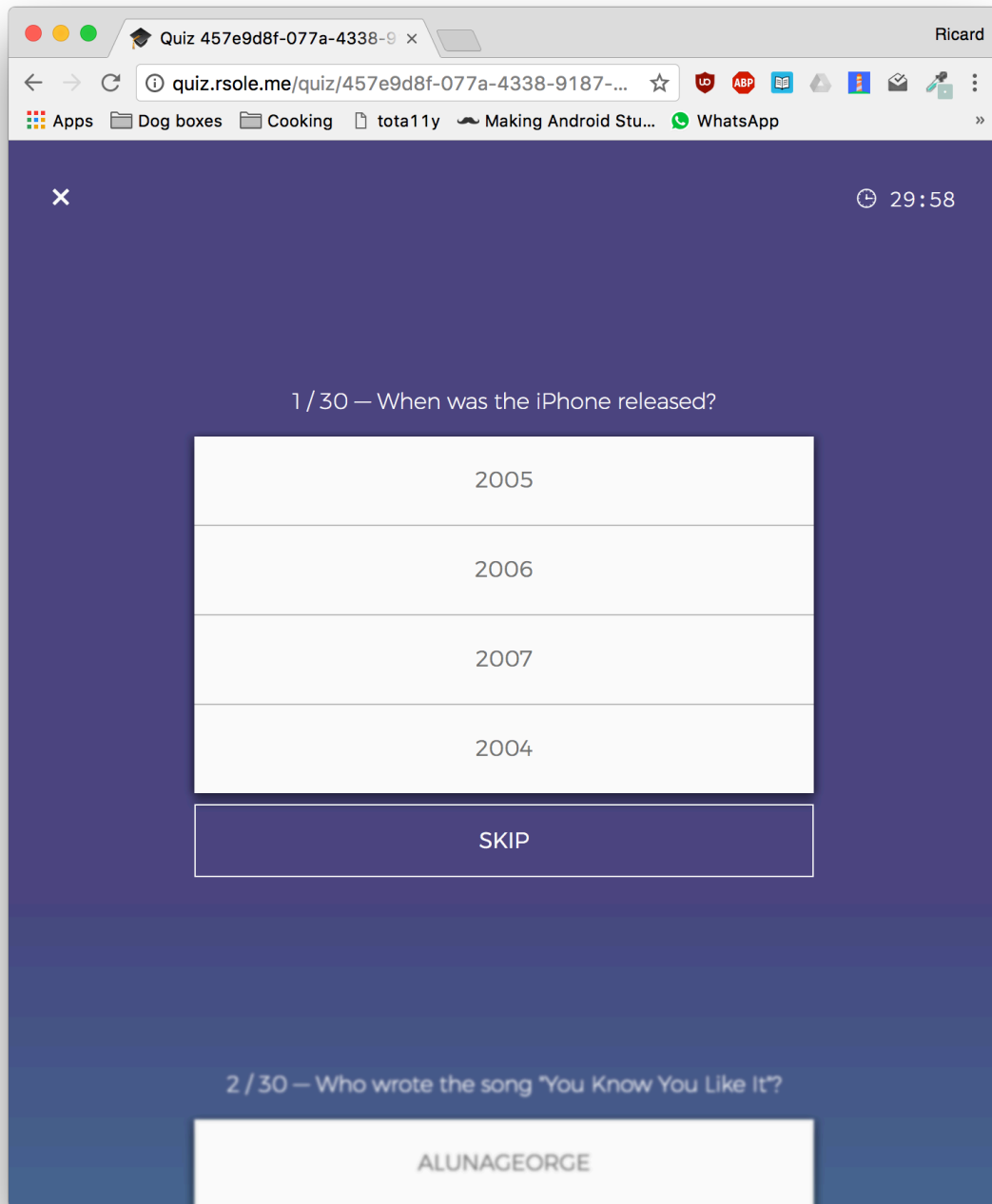Figure 2.1: 10 Questions
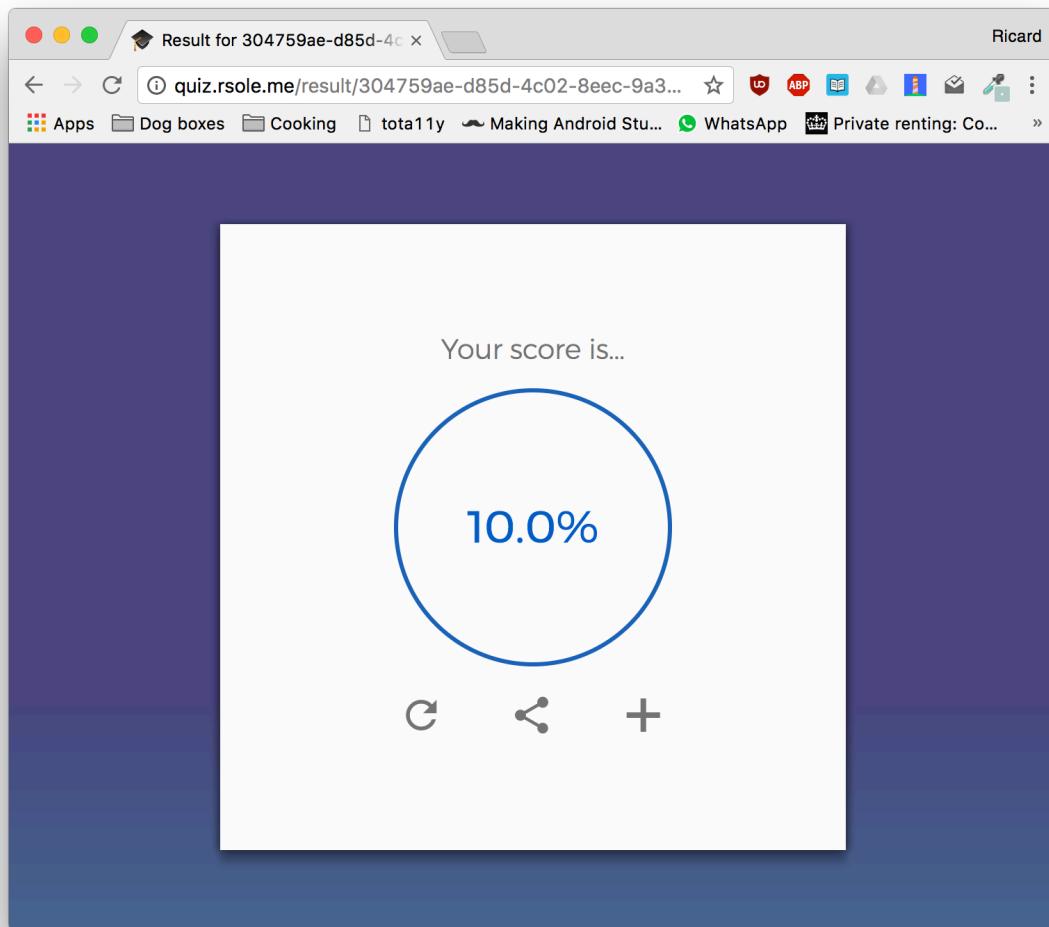
Figure 2.2: 20 Questions

Figure 2.3: 30 Questions

Figure 2.4: Quiz result

Chapter 3

Appendix A: Source Code

3.1 QuizResultsController.java

```java
package controllers;

import com.avaje.ebean.Ebean;
import com.google.inject.Inject;
import models.Quiz;
import models.QuizResult;
import play.data.DynamicForm;
import play.data.FormFactory;
import play.mvc.Controller;
import play.mvc.Result;

import java.util.UUID;

public class QuizResultsController extends Controller {
  @Inject private FormFactory formFactory;

  public Result create() {
    DynamicForm requestData = formFactory.form().bindFromRequest();
    Quiz q = Ebean.find(Quiz.class, UUID.fromString(requestData.get("quiz-id")));
    QuizResult r = new QuizResult();
    r.setScore(q.computeScore(requestData));
    r.setQuiz(q);
    r.save();

    return redirect("/result/" + r.getId());
  }

  public Result show(UUID id) {
    QuizResult r = Ebean.find(QuizResult.class, id);

    return ok(views.html.results.show.render(r));
  }
}
```

3.2 QuizzesController.java

```java
package controllers;

import com.avaje.ebean.Ebean;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.google.inject.Inject;
import models.Quiz;
import play.data.DynamicForm;
import play.data.FormFactory;
import play.mvc.Controller;
import play.mvc.Result;
import services.QuizService;
import views.html.quizzes.form;
import views.html.quizzes.show;

import java.io.IOException;
import java.util.UUID;
import java.util.concurrent.ExecutionException;

public class QuizzesController extends Controller {
  @Inject private FormFactory formFactory;
  private QuizService service = new QuizService();

  public Result form() {
    return ok(form.render());
  }

  public Result create() {
    DynamicForm requestData = formFactory.form().bindFromRequest();
    String amount = requestData.get("questionsAmount");
    String difficulty = requestData.get("difficulty");

    try {
      ObjectMapper mapper = new ObjectMapper();
      String json = service.fetch(amount, difficulty).toCompletableFuture().get();
      Quiz q = mapper.readValue(json, Quiz.class);
      q.setDifficulty(difficulty);
      q.save();
      return redirect("/quiz/" + q.getId());
    } catch (InterruptedException | ExecutionException | IOException e) {
      e.printStackTrace();
    }

    return redirect("/");
  }

  public Result show(UUID id) {
    Quiz q = Ebean.find(Quiz.class, id);

    return ok(show.render(q));
  }
}
```

3.3 Option.java

```java
package models;

import javax.persistence.*;

@Entity
@Table(name = "options")
public class Option {
  @Id private Long id;
  private String text;

  @ManyToOne(cascade = CascadeType.ALL)
  private Question question;

  public Option(String t) {
    this.text = t;
  }

  public Long getId() {
    return id;
  }

  public void setId(Long id) {
    this.id = id;
  }

  public String getText() {
    return text;
  }

  public void setText(String text) {
    this.text = text;
  }

  public Question getQuestion() {
    return question;
  }

  public void setQuestion(Question question) {
    this.question = question;
  }

  @Override
  public String toString() {
    return "Option{" + "text='" + text + '\'' + '}';
  }
}
```

3.4 Question.java

```java
package models;

import com.avaje.ebean.Model;
import com.fasterxml.jackson.annotation.JsonProperty;
```

```java
import javax.persistence.*;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.UUID;

@Entity
@Table(name = "questions")
public class Question extends Model {
  @Id private UUID id;

  @ManyToOne(cascade = CascadeType.ALL)
  private Quiz quiz;

  @JsonProperty("question")
  private String text;

  private String category;
  private String type;
  private String difficulty;

  @JsonProperty("correct_answer")
  private String correctAnswer;

  @JsonProperty("incorrect_answers")
  @OneToMany(cascade = CascadeType.ALL, mappedBy = "question")
  private List<Option> incorrectAnswers;

  public UUID getId() {
    return id;
  }

  public void setId(UUID id) {
    this.id = id;
  }

  public Quiz getQuiz() {
    return quiz;
  }

  public void setQuiz(Quiz quiz) {
    this.quiz = quiz;
  }

  public String getText() {
    return text;
  }

  public void setText(String text) {
    this.text = text;
  }

  public String getCategory() {
    return category.split("\\s|:")[0].toLowerCase();
  }

  public void setCategory(String category) {
    this.category = category;
```

```java
  }

  public String getType() {
    return type;
  }

  public void setType(String type) {
    this.type = type;
  }

  public String getDifficulty() {
    return difficulty;
  }

  public void setDifficulty(String difficulty) {
    this.difficulty = difficulty;
  }

  public String getCorrectAnswer() {
    return correctAnswer;
  }

  public void setCorrectAnswer(String correctAnswer) {
    this.correctAnswer = correctAnswer;
  }

  public List<Option> getIncorrectAnswers() {
    return incorrectAnswers;
  }

  public void setIncorrectAnswers(List<Option> incorrectAnswers) {
    this.incorrectAnswers = incorrectAnswers;
  }

  public List<Option> getOptions() {
    List<Option> os = new ArrayList<>(this.incorrectAnswers);
    os.add(new Option(this.correctAnswer));
    Collections.shuffle(os);

    return os;
  }
}
```

3.5   Quiz.java

```java
package models;

import com.avaje.ebean.Model;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonProperty;
import play.data.DynamicForm;

import javax.persistence.*;
import java.util.List;
import java.util.UUID;

@Entity
```

```java
@Table(name = "quizzes")
@JsonIgnoreProperties({"response_code"})
public class Quiz extends Model {
  @Id private UUID id;

  @OneToMany(mappedBy = "quiz", cascade = CascadeType.ALL)
  @JsonProperty("results")
  private List<Question> questions;

  @OneToMany(mappedBy = "quiz", cascade = CascadeType.ALL)
  private List<QuizResult> quizResults;

  private String difficulty;

  public UUID getId() {
    return id;
  }

  public void setId(UUID id) {
    this.id = id;
  }

  public List<Question> getQuestions() {
    return questions;
  }

  public void setQuestions(List<Question> questions) {
    this.questions = questions;
  }

  public String getDifficulty() {
    return difficulty;
  }

  public void setDifficulty(String difficulty) {
    this.difficulty = difficulty;
  }

  public int computeScore(DynamicForm answers) {
    int score = 0;

    for (Question q : questions) {
      String a = answers.get(q.getId().toString());
      if (a != null && a.equals(q.getCorrectAnswer())) score += 1;
    }

    return score;
  }

  public List<QuizResult> getQuizResults() {
    return quizResults;
  }

  public void setQuizResults(List<QuizResult> quizResults) {
    this.quizResults = quizResults;
  }
}
```

3.6 QuizResult.java

```java
package models;

import com.avaje.ebean.Model;

import javax.persistence.*;
import java.util.UUID;

@Entity
@Table(name = "quiz_results")
public class QuizResult extends Model {
  @Id private UUID id;

  @ManyToOne(cascade = CascadeType.PERSIST)
  private Quiz quiz;

  private int score;

  public UUID getId() {
    return id;
  }

  public void setId(UUID id) {
    this.id = id;
  }

  public Quiz getQuiz() {
    return quiz;
  }

  public void setQuiz(Quiz quiz) {
    this.quiz = quiz;
  }

  public int getScore() {
    return score;
  }

  public double getPercentage() {
    return ((double) score / (double) quiz.getQuestions().size()) * 100;
  }

  public void setScore(int score) {
    this.score = score;
  }
}
```

3.7 QuizService.java

```java
package services;

import play.libs.ws.WS;
import play.libs.ws.WSRequest;
import play.libs.ws.WSResponse;
```

```java
import java.util.concurrent.CompletionStage;

public class QuizService {
  public CompletionStage<String> fetch(String amount, String difficulty) {
    return request(amount, difficulty).thenApply((WSResponse r) -> r.asJson().toString());
  }

  private CompletionStage<WSResponse> request(String amount, String difficulty) {
    final String HOST = "https://opentdb.com/api.php";
    final WSRequest req =
        WS.url(HOST)
            .setQueryParameter("amount", amount)
            .setQueryParameter("type", "multiple")
            .setContentType("application/json");

    if (!difficulty.equals("mixed")) {
      return req.setQueryParameter("difficulty", difficulty).get();
    }

    return req.get();
  }
}
```